Universal Multi-Party Poisoning Attacks

Saeed Mahloujifar
Mohammad Mahmoody
Ameer Mohammed
Multi-Party Learning

Distributions: $D_1, D_n$

Data Providers: $P_1, P_n$

Model: $G$
Multi-Party Learning (Round $j$)

Distributions | Data Providers
--- | ---
$D_1$ | $P_1$

$D_i$ | $P_i$

$D_n$ | $P_n$

$G$

Model$_{j-1}$
Multi-Party Learning (Round j)

Distributions

$D_1$

$D_i$

$D_n$

Data Providers

$P_1$

$P_i$

$P_n$

Model_{j-1}

$G$
Multi-Party Learning (Round j)

Distributions

\( D_1 \rightarrow P_1 \)

\( D_i \rightarrow P_i \)

\( D_n \rightarrow P_n \)

Data Providers

Model_{j-1}

\( G \)

\( d_j \rightarrow u_j \)
Multi-Party Learning (Round j)

Distributions

\[ D_1 \rightarrow P_1 \]
\[ D_i \rightarrow d_j \rightarrow P_i \]
\[ D_n \rightarrow P_n \]

Data Providers

Model_{j-1} \rightarrow G \rightarrow u_j \rightarrow P_i \rightarrow d_j \rightarrow D_i

Model_j
Multi-Party Learning (Round j)

Distributions

\[ D_1 \]

\[ D_i \]

\[ D_n \]

Data Providers

\[ P_1 \]

\[ P_i \]

\[ P_n \]

Model \( j \)

Model \( j \)

Model \( j \)

Model \( j \)

Model \( j \)

Model \( j \)

Model \( j-1 \)

G
Poisoning in Multi-Party Learning

An adversary (partially) controls a number of data providers

\[ D_1, D_i, D_n \] → \[ P_1, P_i, P_n \]

\[ G \] → Model
$(k, q)$-Poisoning Attack Model

$k$ (out of $n$) of the parties become corrupted

Each corrupted party $P_i$ samples from a different distribution $D_i$

$$d(D_i, D_i) \leq q$$

$k = n \rightarrow q$-Tampering [ACMPS14] [MM17] [MM18]

$q = 1 \rightarrow$ Static Corruption in MPC (crypto)
What is the inherent power of \((k, q)\)-poisoning adversaries against Multi-party Learning?
Main Theorem: Power of \((k, q)\)-Poisoning

Let \(B\) be a bad property of the model \(M\)
  
  - E.g. \(B(M) = 1\) if \(M\) misclassified an specific instance \(x\)

For any \(n\)-party learning protocol there is a \((k, q)\)-poisoning adversary that increases \(\Pr[B]\) from

\[
\epsilon \rightarrow \epsilon^{1 - \frac{kq}{n}}
\]
Main Theorem: Power of \((k, q)\)-Poisoning

Let \(B\) be a bad property of the model \(M\)

- E.g. \(B(M) = 1\) if \(M\) misclassified an specific instance \(x\)

For any \(n\)-party learning protocol there is a \((k, q)\)-poisoning adversary that increases \(\Pr[B]\) from

\[
\epsilon \rightarrow \epsilon^{1 - \frac{kq}{n}}
\]

<table>
<thead>
<tr>
<th>(\Pr[B]) Before attack</th>
<th>(q)</th>
<th>(k)</th>
<th>(\Pr[B]) after attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>1/2</td>
<td>(n/2)</td>
<td>11%</td>
</tr>
<tr>
<td>5%</td>
<td>1/2</td>
<td>(n)</td>
<td>22%</td>
</tr>
<tr>
<td>5%</td>
<td>1</td>
<td>(n/2)</td>
<td>22%</td>
</tr>
</tbody>
</table>
Features of Attack

• **Universal**: provably work against any learning protocol
  • In contrast with: [Bagdasaryan et al 2018; Bhagoji et al. 2018]

• **Clean label**: Only uses correct labels
  • Similar to: [M et al 2017; Shafahi et al 2018]

• **Polynomial time**
  • Similar to: [M and Mahmoudy 2019]
Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
Ideas Behind Attack

• Main Idea: Treat protocol as random process and run a biasing attack
  • The bad property is a function over the random process
  • We want to bias that function, similar to attacks in coin tossing
• New biasing model: Generalized $p$-Tampering.
Ideas Behind Attack

• Main Idea: Treat protocol as random process and run a biasing attack
  • The bad property is a function over the random process
  • We want to bias that function, similar to attacks in coin tossing

• New biasing model: Generalized $p$-Tampering.

Let $f : (U_1, ..., U_n) \rightarrow \{0,1\}$
Ideas Behind Attack

• Main Idea: Treat protocol as random process and run a biasing attack
  • The bad property is a function over the random process
  • We want to bias that function, similar to attacks in coin tossing

• New biasing model: Generalized $p$-Tampering.

Let $f : (U_1, ..., U_n) \rightarrow \{0,1\}$
Input blocks $u_1, u_2, ... u_n$ are sampled one-by one in online way:
Ideas Behind Attack

• Main Idea: Treat protocol as random process and run a biasing attack
  • The bad property is a function over the random process
  • We want to bias that function, similar to attacks in coin tossing

• New biasing model: Generalized \( p \)-Tampering.

Let \( f : (U_1, ..., U_n) \rightarrow \{0,1\} \)

Input blocks \( u_1, u_2, ... u_n \) are sampled one-by-one in online way:

\[
u_i = \begin{cases} 
U_i & \text{with marginal probability } 1 - p \\
\text{with marginal probability } p & \text{with marginal probability } p
\end{cases}
\]
Ideas Behind Attack

• Main Idea: Treat protocol as random process and run a biasing attack
  • The bad property is a function over the random process
  • We want to bias that function, similar to attacks in coin tossing

• New biasing model: Generalized $p$-Tampering.

Let $f : (U_1, ..., U_n) \rightarrow \{0,1\}$

Input blocks $u_1, u_2, ... u_n$ are sampled one-by-one in online way:

$$u_i = \begin{cases} U_i & \text{with marginal probability } 1 - p \\ \text{devil} & \text{with marginal probability } p \end{cases}$$

Our generalized $p$-tampering attack based on Ideas in coin tossing attacks [BOL89,IH14]
Summary

We show Poisoning attacks against multi-party learning protocols:

- **Universal**: Provably apply to any multi-party learning protocol
- **Clean label**: Only uses samples with correct labels
- Run in **polynomial time**
- Increase the probability of **any chosen bad property**